

Assignment Day 15

Task 1:

Create a Scala application to find the GCD of two numbers

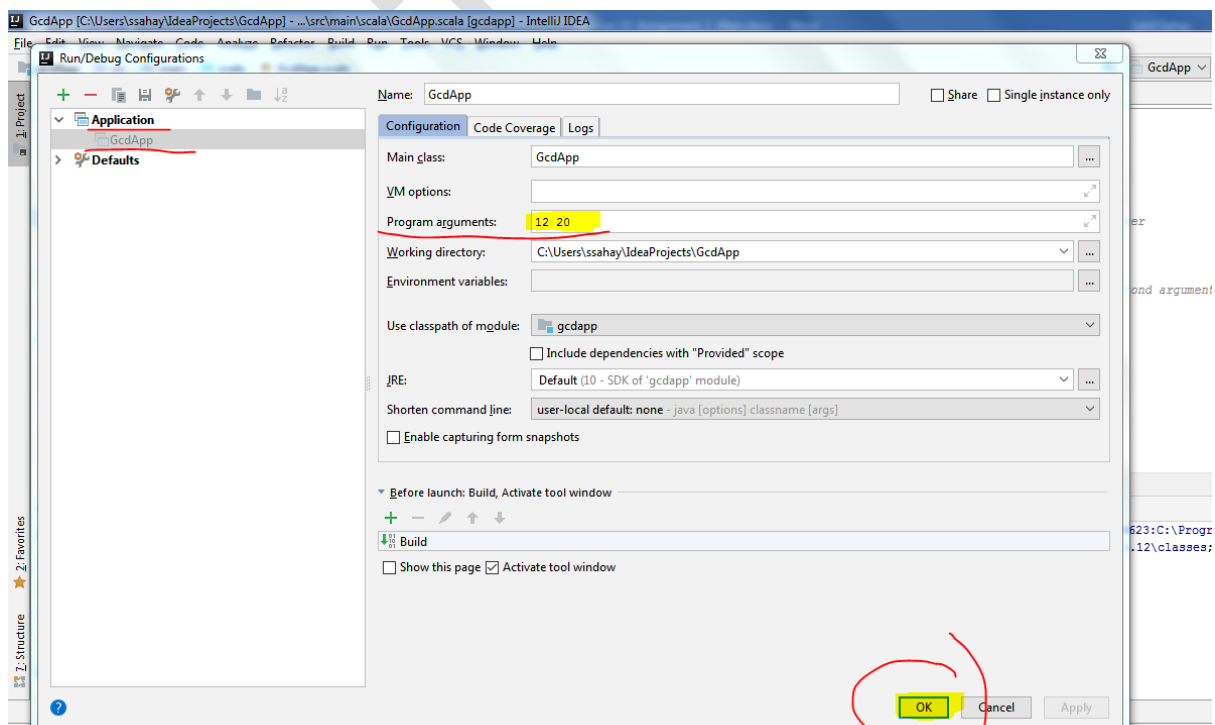
Ans:

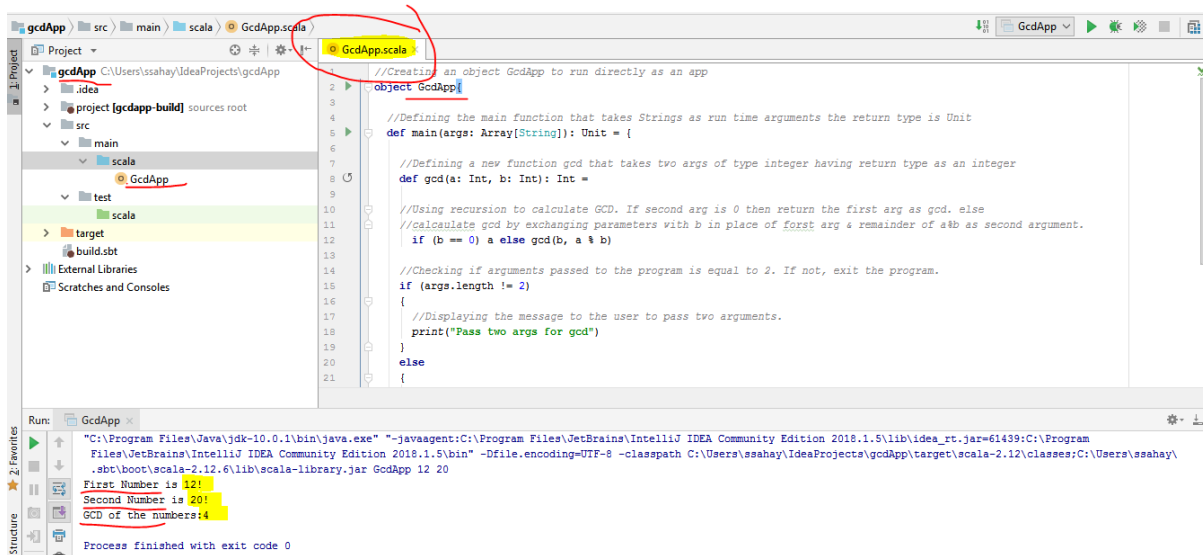
Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

Program Set up:

Goto Run -> Edit Configurations -> under Application(GcdApp) -> Program arguments -> Pass the input arguments.

ScreenShot:





Task 2:

Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.

Write a Scala application to find the Nth digit in the sequence.

- Write the function using standard for loop.
- Write the function using recursion.

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

ScreenShot:

```
1 //Created an object singleton object fibApp to run this app directly.
2 object fibApp {
3     //Declaring & defining a new function fib passing first arg as 0, second arg as 1, count as 2 & returning a list
4     def fib(a: Int = 0, b: Int = 1, count: Int = 2): List[Int] = {
5         //To calculate the next value we add first and second number
6         val c = a + b
7         //Stopping criteria, send back a list containing the latest value
8         if (count >= 12) {
9             List(c)
10        }
11        //If this is the first iteration create the first few fibonacci numbers, and make a recursive call
12        //Adding one list to another is done using the ++ function
13        else if (a == 0 && b == 1) {
14            List(a, b, c) ++ fib(b, c, count + 1)
15        }
16        //If this wasn't the first iteration, just add the latest element at the beginning of the list and make the recursive call
17        else {
18            c :: fib(b, c, count + 1)
19        }
20    }
21    //Defining the main function that is the starting point of this application. Calling fib() function & the return type is Unit
22    fibApp > fib(...)

```

Run: fibApp

```
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\lib\idea_rt.jar=65354:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\ssahay\IdeaProjects\fibApp\target\scala-2.12\classes;C:\Users\ssahay\IdeaProjects\fibApp\target\scala-2.12\lib\scala-library.jar fibApp
List(0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144)
Process finished with exit code 0

```

Task 3

Find square root of number using Babylonian method.

1. Start with an arbitrary positive start value x (the closer to the root, the better).
2. Initialize $y = 1$.
3. Do following until desired approximation is achieved.
 - a) Get the next approximation for root using average of x and y
 - b) Set $y = n/x$

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

ScreenShot:

```
1 | Stream.cons(x, squareRoot(0.5*(x + n / x), n))
2 | // to find squareRoot of x, we will add x with n(root number) and multiply the result //with 0.5
3 | squareRoot(1, n)
4 | squareRoot(2) // streaming 5 iterations to find squareRoot
5 | val iterations = 5
6 | squareRoot(2)(iterations-1)
7 | squareRoot(2).take(iterations).toList
8 |
9 |
10 | def squareRoot(n: Double): Double = {
11 |   var x: Double = n
12 |   var y: Double = 1
13 |   val e: Double = 0.000001
14 |   while (x-y > e) {
15 |     x = (x+y)/2
16 |     y = n/x // while loop to find the squareRoot
17 |   }
18 |   x
19 | }
20 | //Defining the main function that is the starting point of this application. Calling fib() function & the return type is Unit
21 | def main(args: Array[String]): Unit = {
22 |   // Calling the "squareRoot" function to execute the square root of 12.1234556
23 |   println(squareRoot(12.8567494))
24 | }
25 |
26 | squareRoot > main(args: Array[String])
```

Run: squareRoot

```
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\lib\idea_rt.jar=59710:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\ssahay\IdeaProjects\squareRoot\target\scala-2.12\classes;C:\Users\ssahay\IdeaProjects\squareRoot\target\scala-2.12\lib\scala-library.jar squareRoot
3.5856310268160447
Process finished with exit code 0
```

```
14 | def squareRoot(n: Double): Double = {
15 |   var x: Double = n
16 |   var y: Double = 1
17 |   val e: Double = 0.000001
18 |   while (x-y > e) {
19 |     x = (x+y)/2
20 |     y = n/x // while loop to find the squareRoot
21 |   }
22 |   x
23 | }
24 | //Defining the main function that is the starting point of this application. Calling fib() function & the return type is Unit
25 | def main(args: Array[String]): Unit = {
26 |   // Calling the "squareRoot" function to execute the square root of 12.1234556
27 |   println(squareRoot(196))
28 | }
29 |
squareRoot > main(args: Array[String])
```

Run: squareRoot

```
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\lib\idea_rt.jar=58212:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\ssahay\IdeaProjects\squareRoot\target\scala-2.12\classes;C:\Users\ssahay\IdeaProjects\squareRoot\target\scala-2.12\lib\scala-library.jar squareRoot
14.000000310617537
```

End
